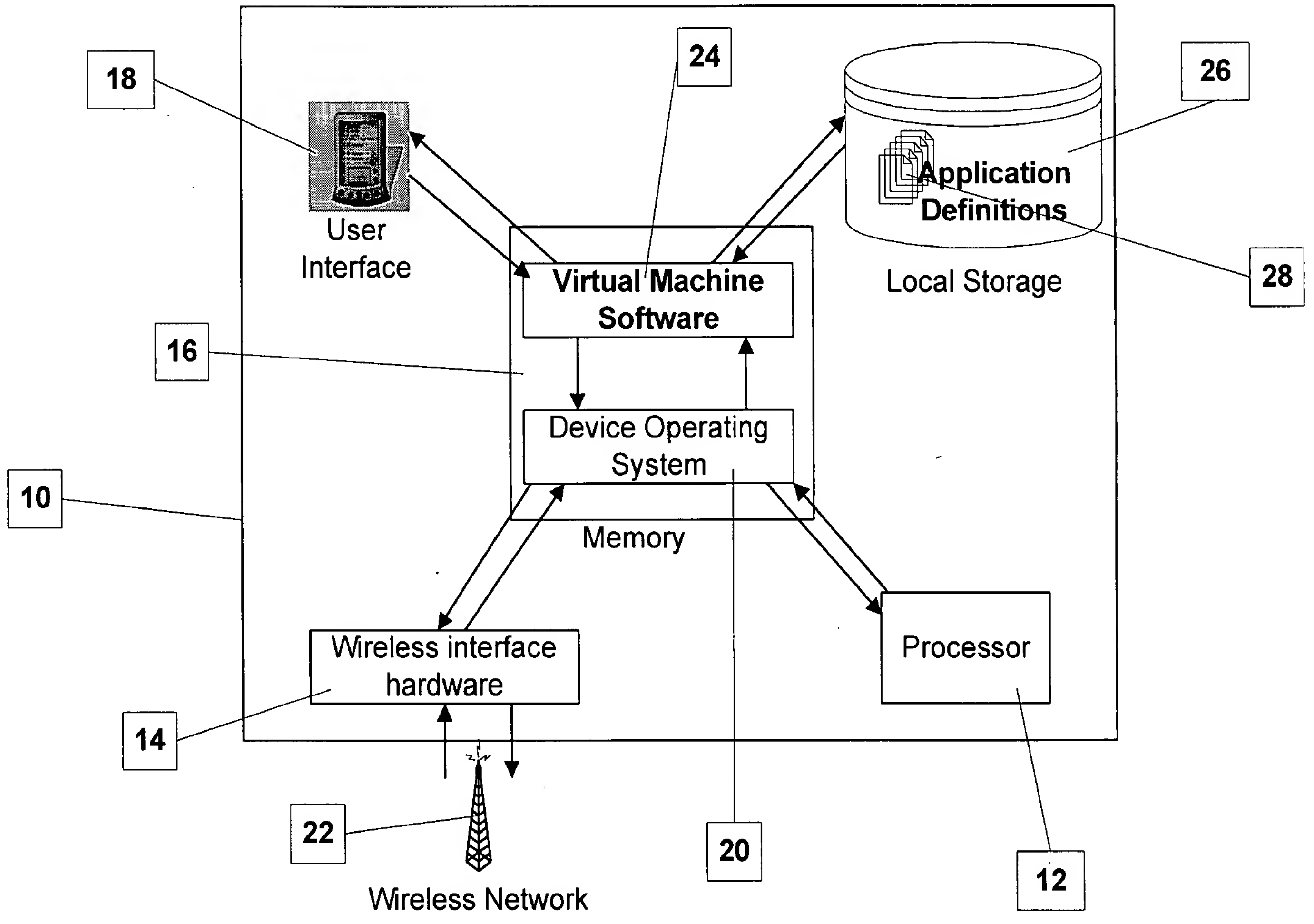
FIG. 1



18
User Interface
24
Virtual Machine Software
26
Application Definitions
Local Storage
28
16
Device Operating System
Memory
10
Wireless interface hardware
Processor
14
22
Wireless Network
20
12

FIG. 2

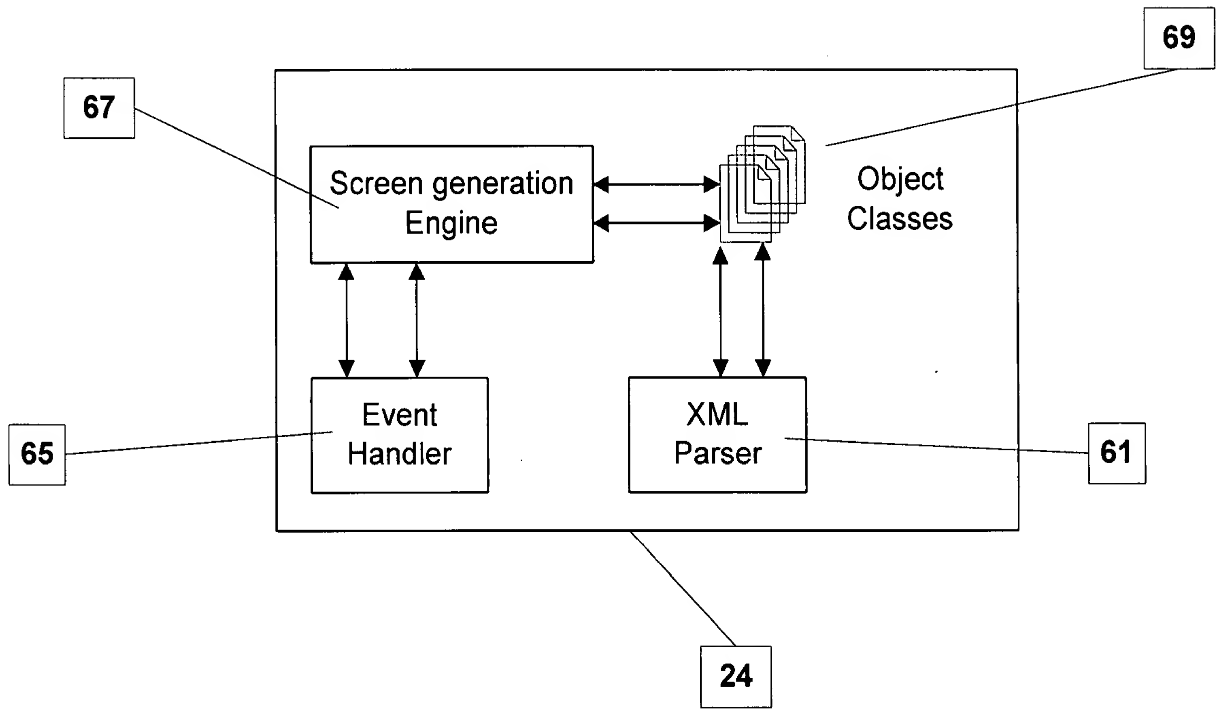

67
Screen generation Engine
Object Classes
69
Event Handler
65
XML Parser
61
24

FIG. 3

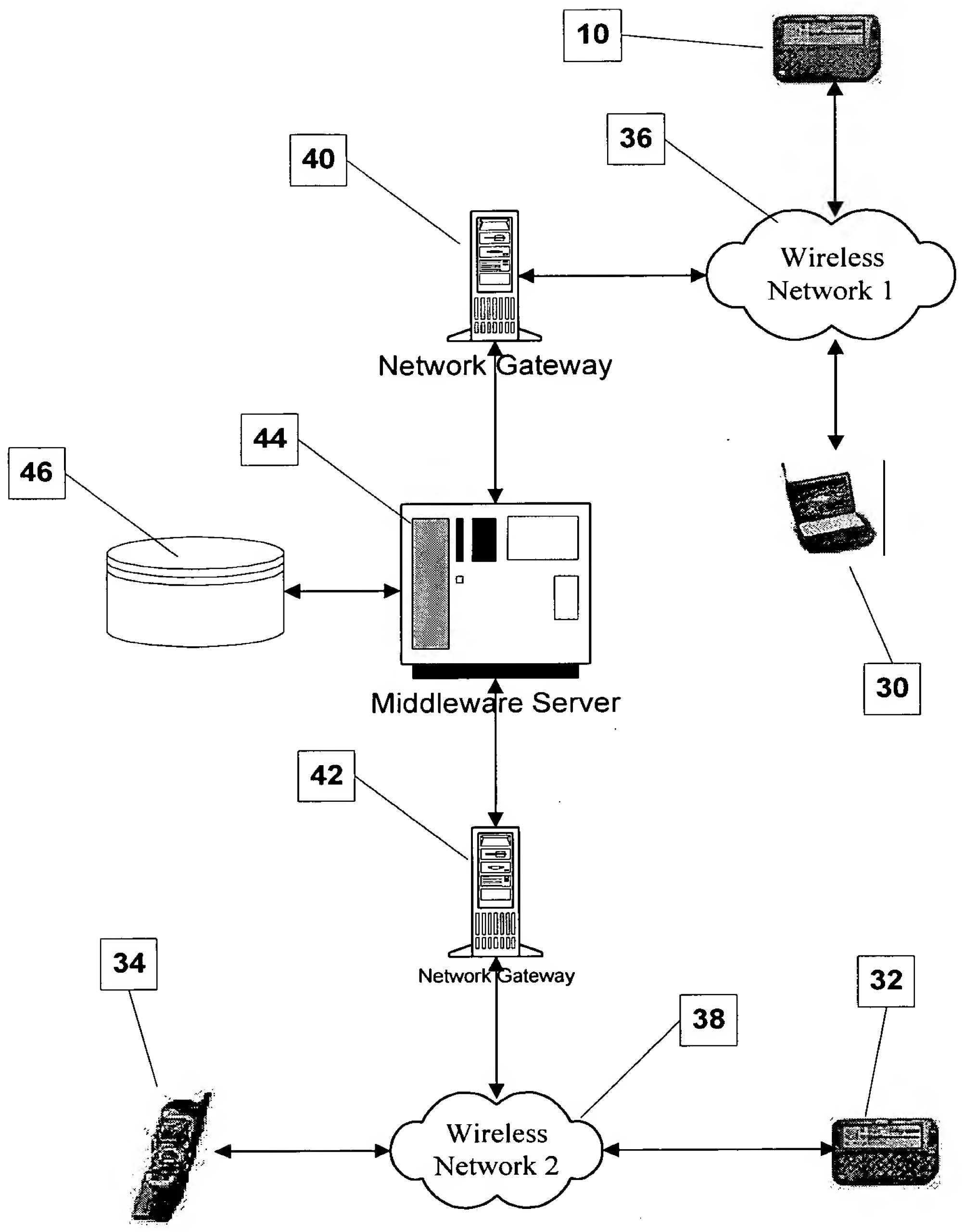
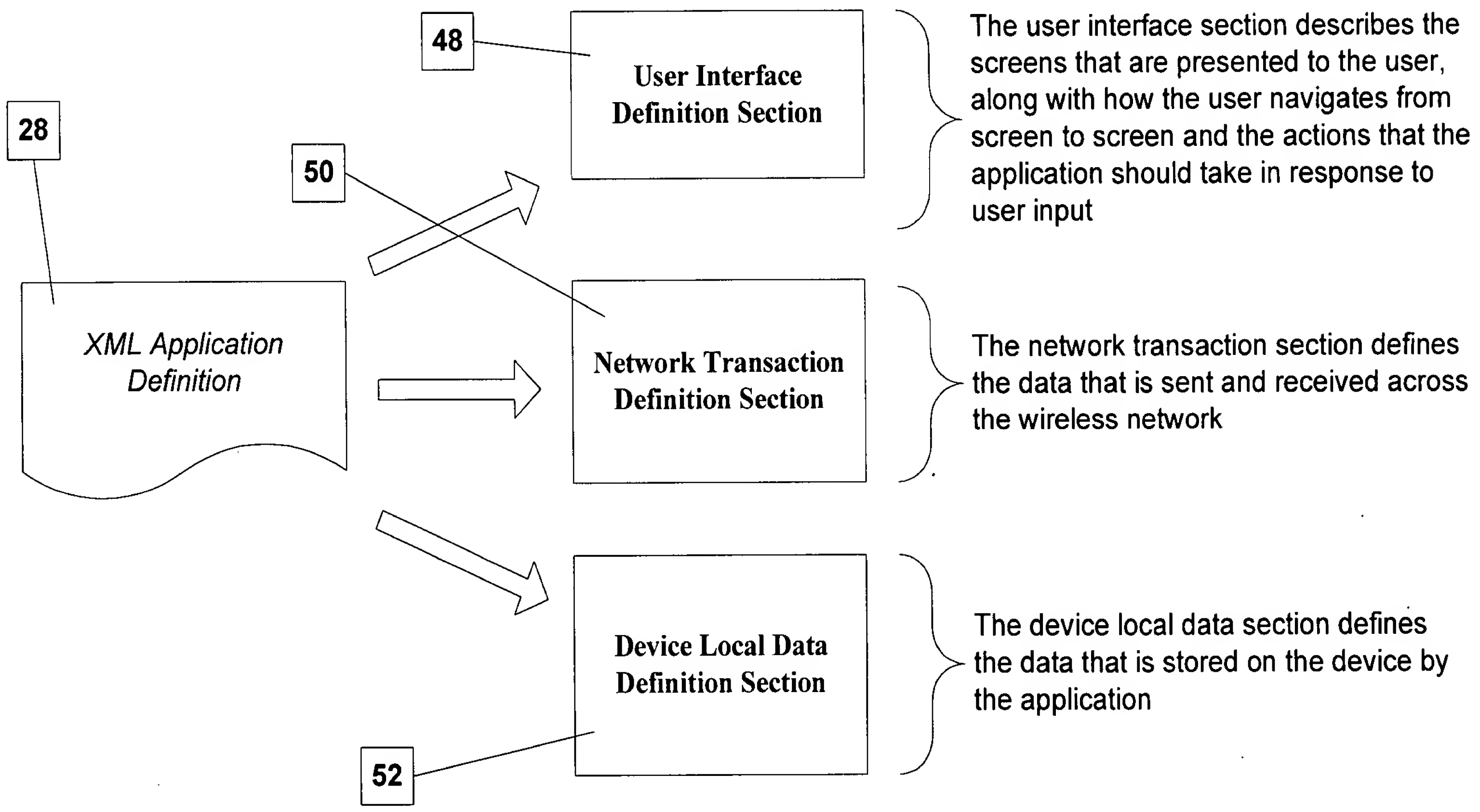

10
36
40
Wireless Network 1
Network Gateway
44
46
Middleware Server
30
42
Network Gateway
34
38
32
Wireless Network 2

FIG. 4


28
XML Application Definition
48
User Interface Definition Section
50
Network Transaction Definition Section
52
Device Local Data Definition Section
The user interface section describes the screens that are presented to the user, along with how the user navigates from screen to screen and the actions that the application should take in response to user input
The network transaction section defines the data that is sent and received across the wireless network
The device local data section defines the data that is stored on the device by the application

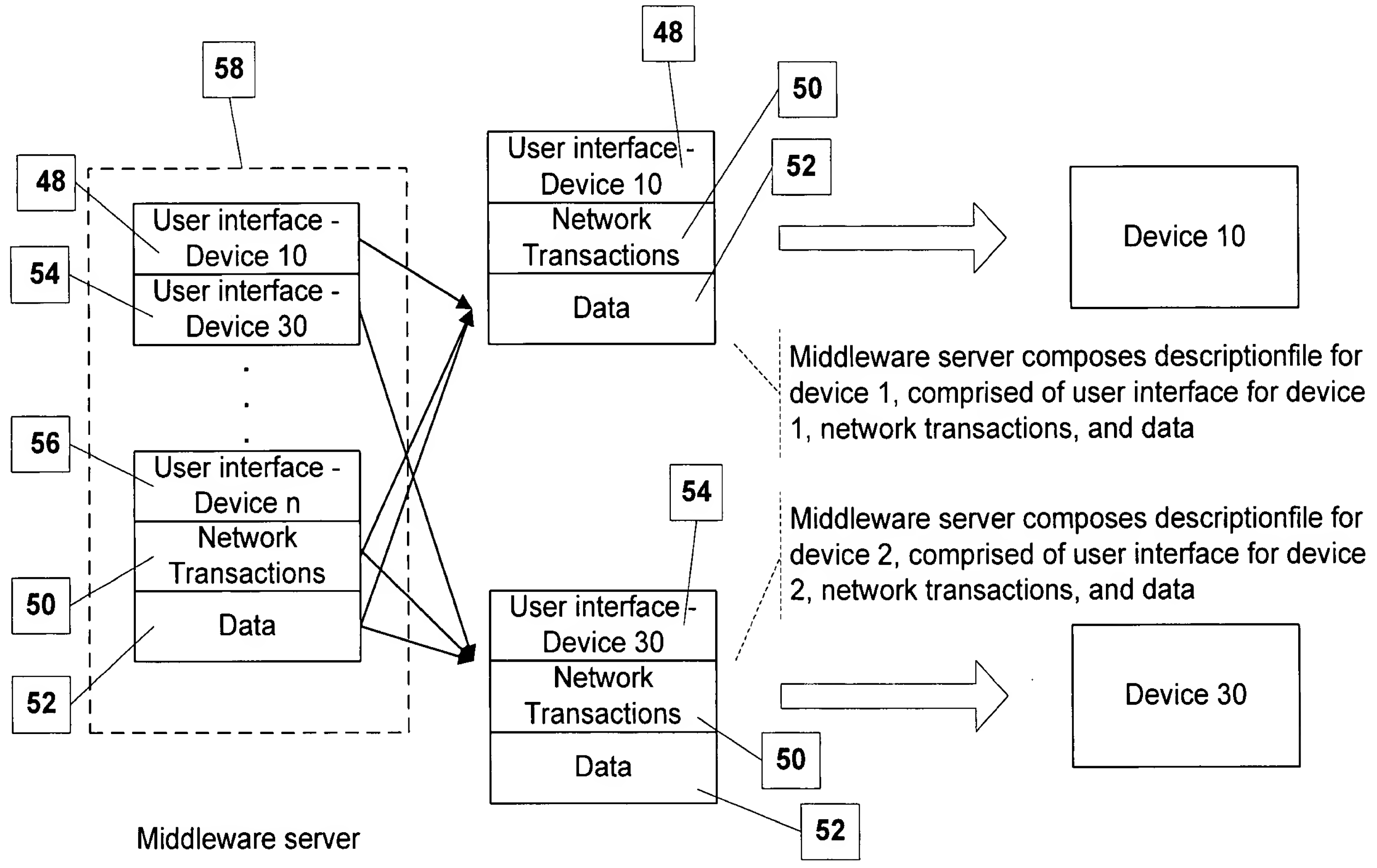
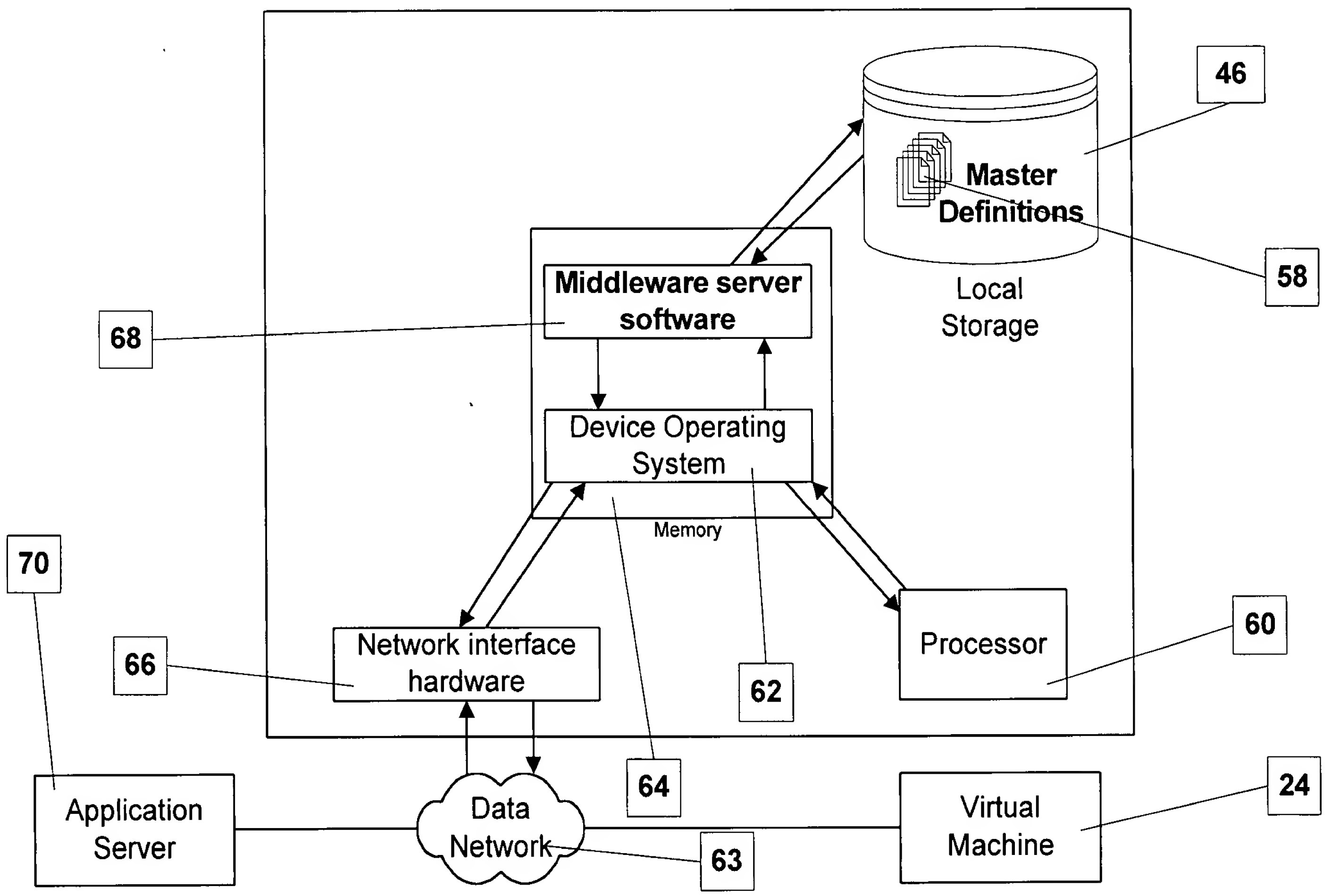

58
48
54
56
50
52
User interface - Device 10
User interface - Device 30
User interface - Device n
Network Transactions
Data
48
50
52
User interface - Device 10
Network Transactions
Data
Device 10
Middleware server composes descriptionfile for device 1, comprised of user interface for device 1, network transactions, and data
54
User interface - Device 30
Network Transactions
Data
50
52
Middleware server composes descriptionfile for device 2, comprised of user interface for device 2, network transactions, and data
Device 30
Middleware server

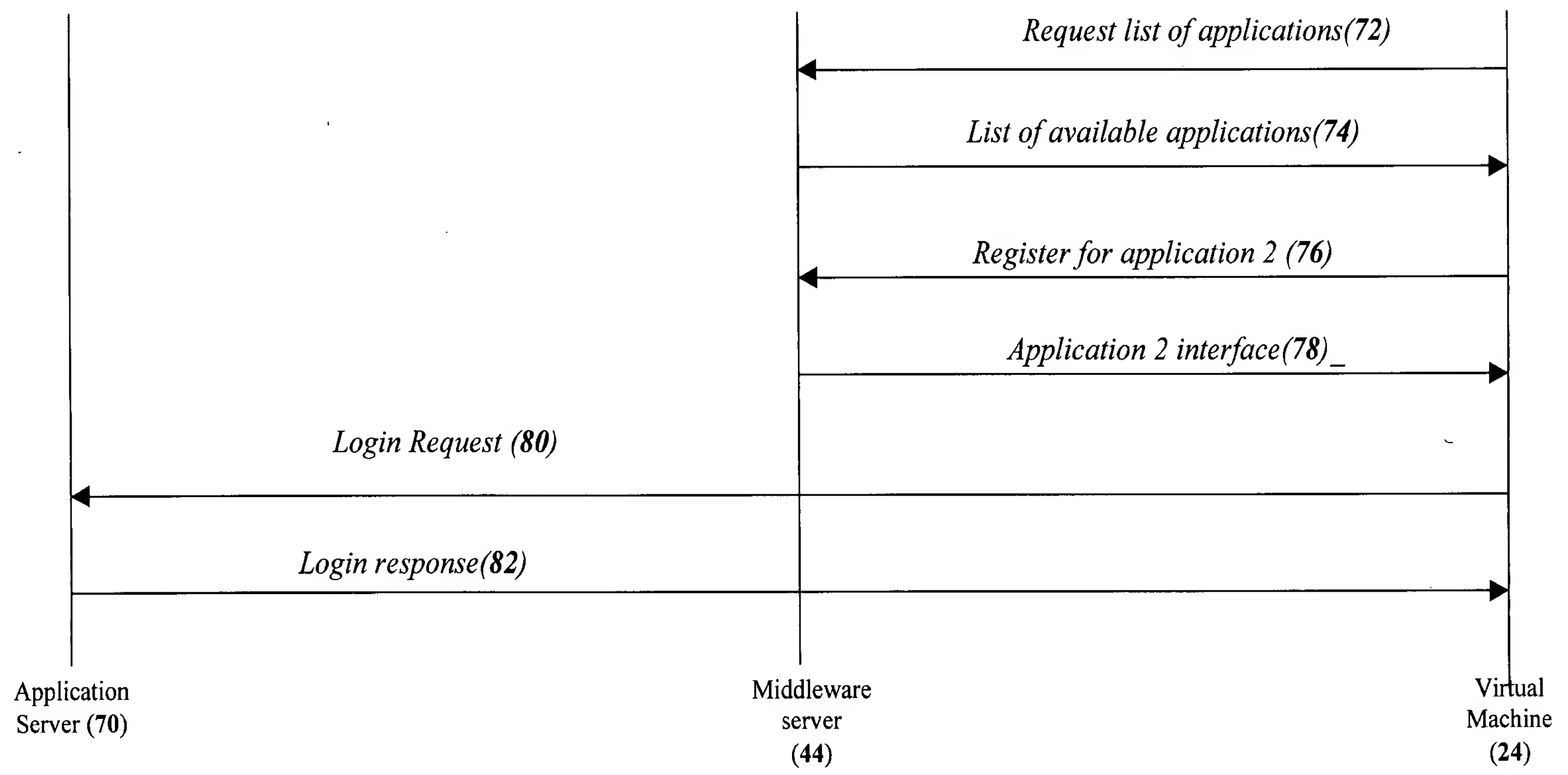
FIG. 5

FIG. 6



46
Master Definitions
58
Local Storage
68
Middleware server software
Device Operating System
Memory
70
66
Network interface hardware
62
Processor
60
64
Application Server
Data Network
63
Virtual Machine
24

FIG. 7



Request list of applications(72)
List of available applications(74)
Register for application 2 (76)
Application 2 interface(78)_
Login Request (80)
Login response(82)
Application Server (70)
Middleware server (44)
Virtual Machine (24)

FIG. 8

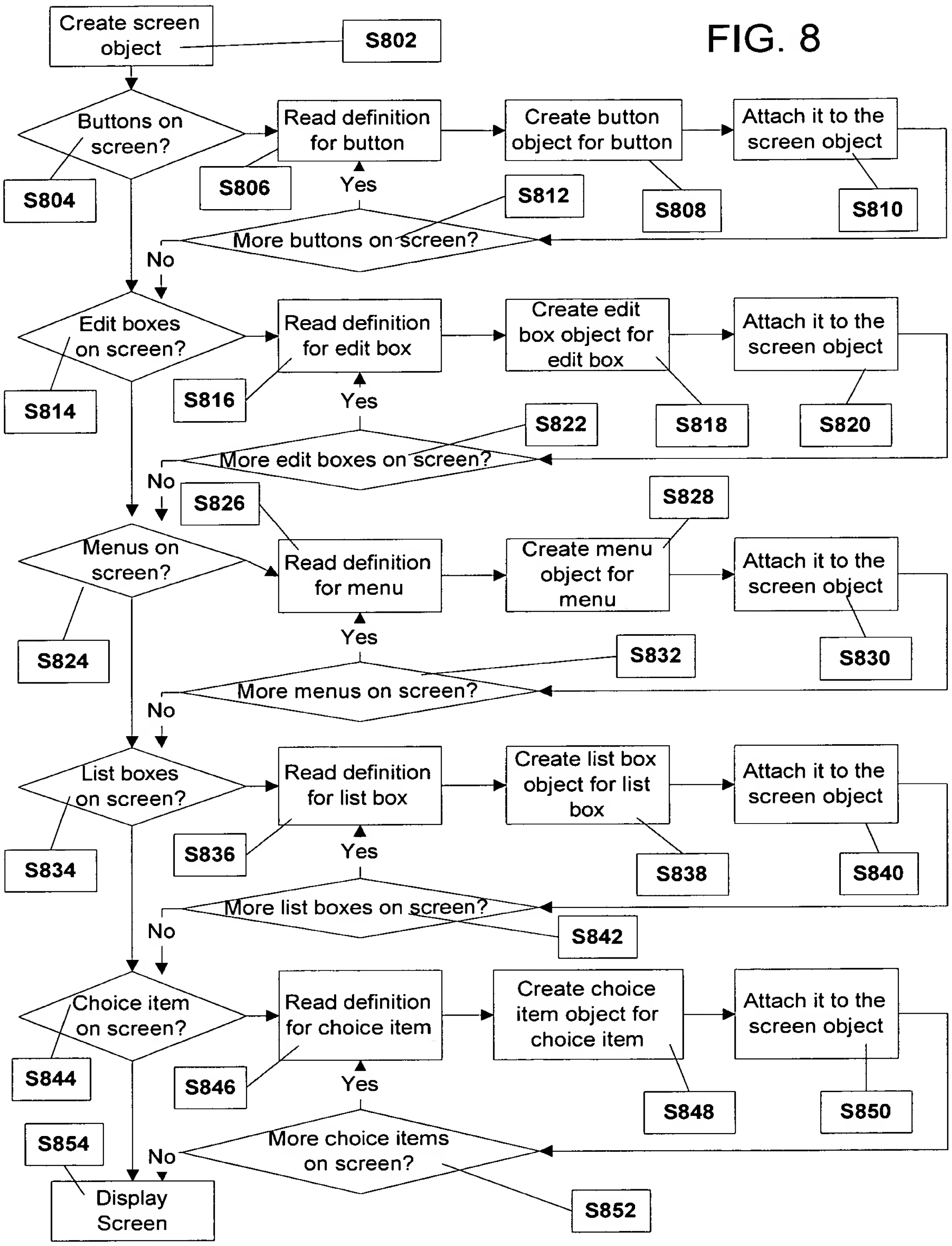

Create screen object
S802
Buttons on screen?
S804
Read definition for button
S806
Create button object for button
S808
Attach it to the screen object
S810
Yes
More buttons on screen?
S812
No
Edit boxes on screen?
S814
Read definition for edit box
S816
Create edit box object for edit box
S818
Attach it to the screen object
S820
Yes
More edit boxes on screen?
S822
No
S826
S828
Menus on screen?
S824
Read definition for menu
Create menu object for menu
Attach it to the screen object
S830
Yes
More menus on screen?
S832
No
List boxes on screen?
S834
Read definition for list box
S836
Create list box object for list box
S838
Attach it to the screen object
S840
Yes
More list boxes on screen?
S842
No
Choice item on screen?
S844
Read definition for choice item
S846
Create choice item object for choice item
S848
Attach it to the screen object
S850
Yes
More choice items on screen?
S852
No
S854
Display Screen

FIG. 9

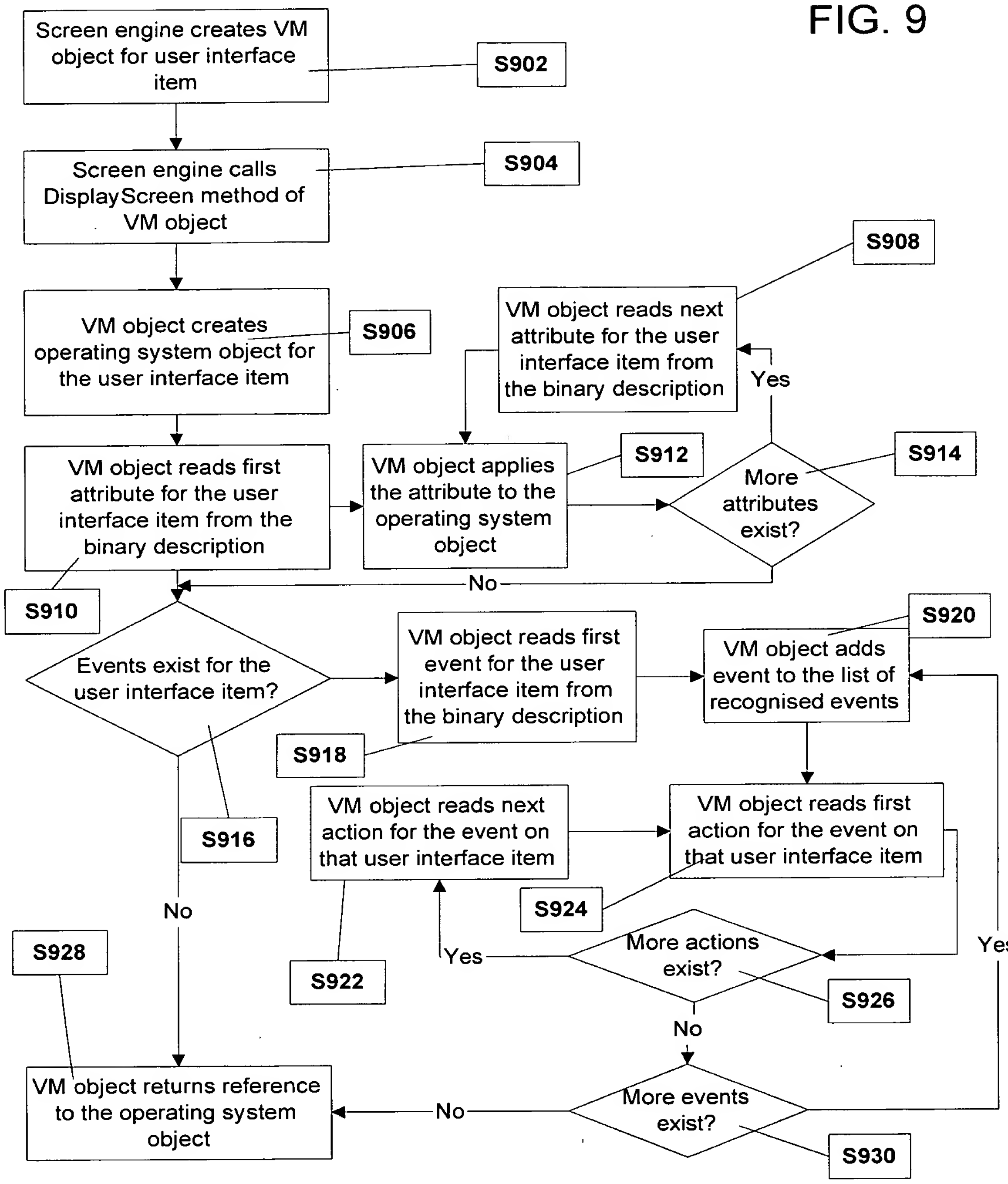
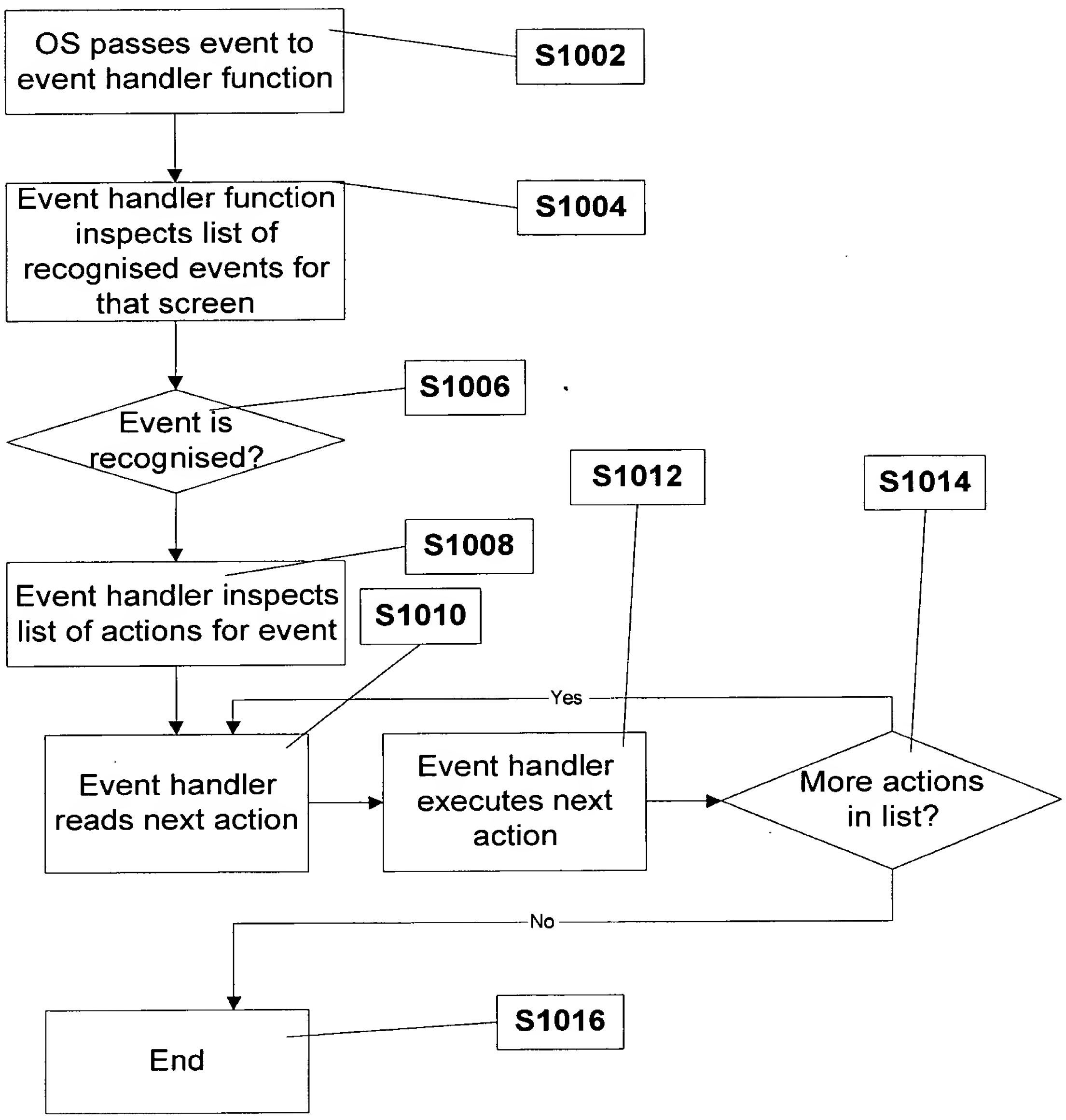

Screen engine creates VM object for user interface item
S902
Screen engine calls DisplayScreen method of VM object
S904
S908
VM object reads next attribute for the user interface item from the binary description
VM object creates operating system object for the user interface item
S906
Yes
VM object reads first attribute for the user interface item from the binary description
VM object applies the attribute to the operating system object
S912
More attributes exist?
S914
S910
No
S920
Events exist for the user interface item?
VM object reads first event for the user interface item from the binary description
VM object adds event to the list of recognised events
S918
VM object reads next action for the event on that user interface item
VM object reads first action for the event on that user interface item
S916
S924
No
Yes
S928
Yes
More actions exist?
S922
S926
No
VM object returns reference to the operating system object
No
More events exist?
S930

FIG. 10



OS passes event to event handler function
S1002
Event handler function inspects list of recognised events for that screen
S1004
S1006
Event is recognised?
S1012
S1014
S1008
Event handler inspects list of actions for event
S1010
Yes
Event handler reads next action
Event handler executes next action
More actions in list?
No
End
S1016

FIG. 11

```
<ARML>
        <HEAD>…</HEAD>
        <SYS>
                <QUERY>
                        <PLATFORMS>
                                <PLATFORM>WinCE</PLATFORM>
                        </PLATFORMS>
                </REG>
        </SYS>
</ARML>
```

72

```
<ARML>
        <HEAD>…</HEAD>
        <SYS>
                <QUERYRESP>
                        <APP>Order Entry</APP>
                        <APP>Helpdesk</APP>
                        <APP>Engineer Dispatch</APP>
                </QUERYRESP>
        </SYS>
</ARML>
```

74

```
<ARML>
        <HEAD>…</HEAD>
        <SYS>
                <REG TYPE="ADD">
                        <CLIENTID>SUNTRESS</CLIENTID>
                        <MOBILEID>867452</MOBILEID>
                        <NEWMOBILEID>268625</NEWMOBILEID>
                        <PLATFORMS>
                                <PLATFORM>WinCE</PLATFORM>
                        </PLATFORMS>
                </REG>
        </SYS>
</ARML>
```

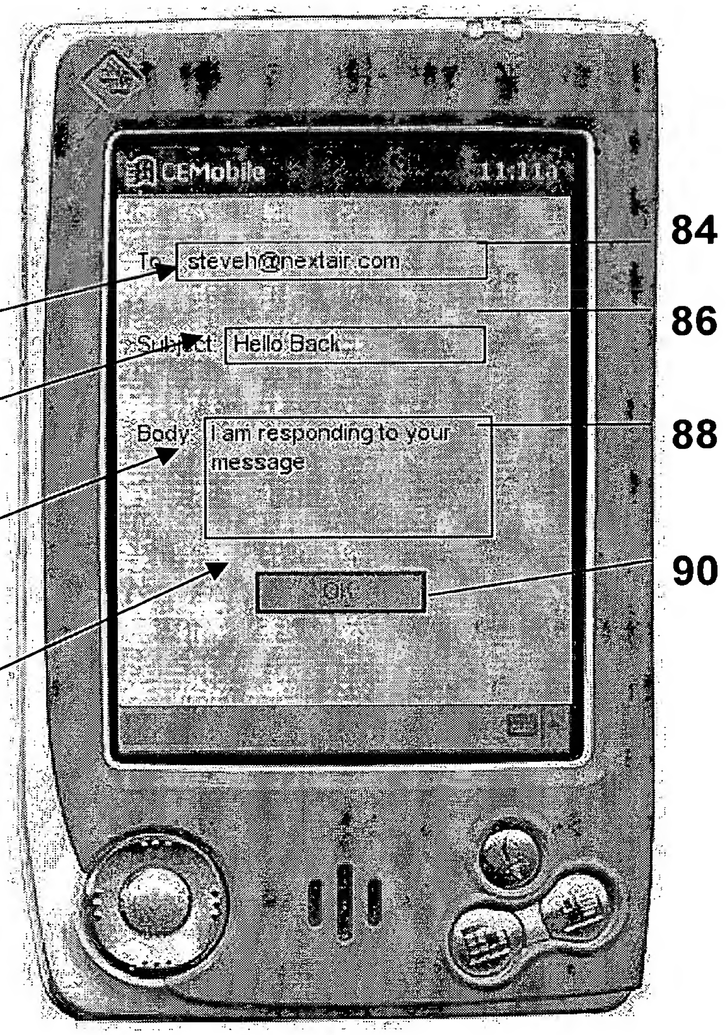
76

```
<ARML>
        <HEAD>…</HEAD>
        <SYS>
                <REGCONFIRM TYPE="ADD">
                        <MOBILEID>268625</MOBILEID>
                        <VALUE>CONFIRM</VALUE>
                        <INTERFACE>
                            <BUTTONS NUM="1">
                                <BTN NAME="OK" CAPTION="Send" INDEX="0">
                                </BTN>
                            </BUTTONS>
                            <EDITBOXES NUM="3">
                                <EB NAME="To" INDEX="1"></EB>
                                <EB NAME="Subject" INDEX="2"></EB>
                                <EB NAME="Body" INDEX="3"></EB>
                            </EDITBOXES>
                        </INTERFACE>
                </REGCONFIRM>
        </SYS>
</ARML>
```

78

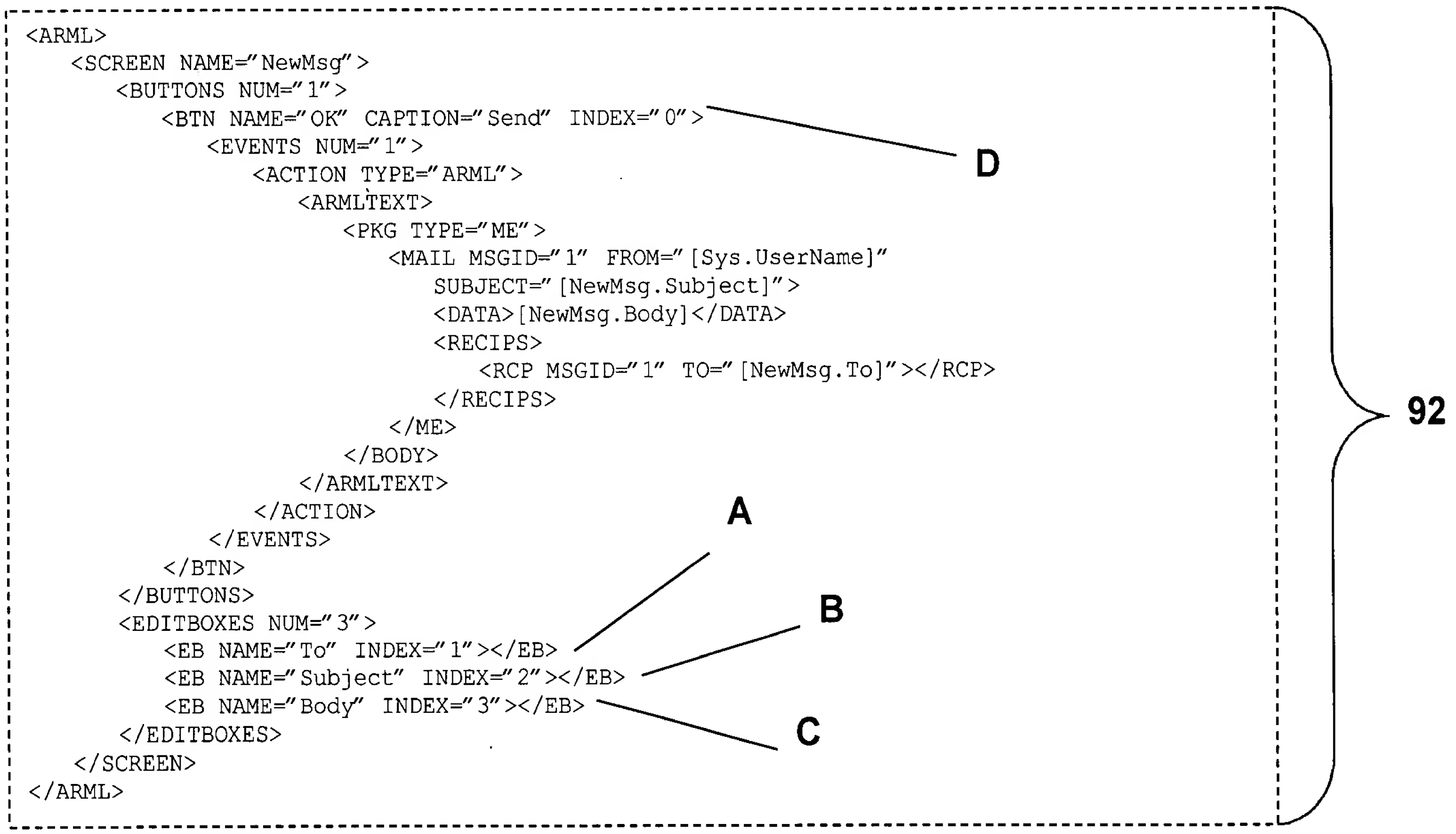
FIG. 12

```
<EDITBOXES NUM="3">
        <EB NAME="To"  INDEX="1"></EB>

        <EB NAME="Subject"  INDEX="2"></EB>

        <EB NAME="Body"  INDEX="3"></EB>
</EDITBOXES>

<BUTTONS NUM="1">
        <BTN NAME="OK"  CAPTION="Send"
INDEX="0"></BTN>
</BUTTONS>
```



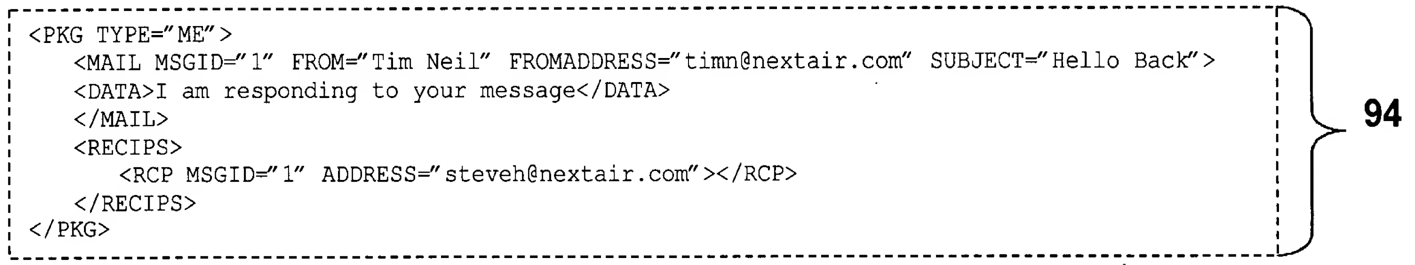
To: steveh@nextair.com
Subject: Hello Back...
Body: I am responding to your message

FIG. 13



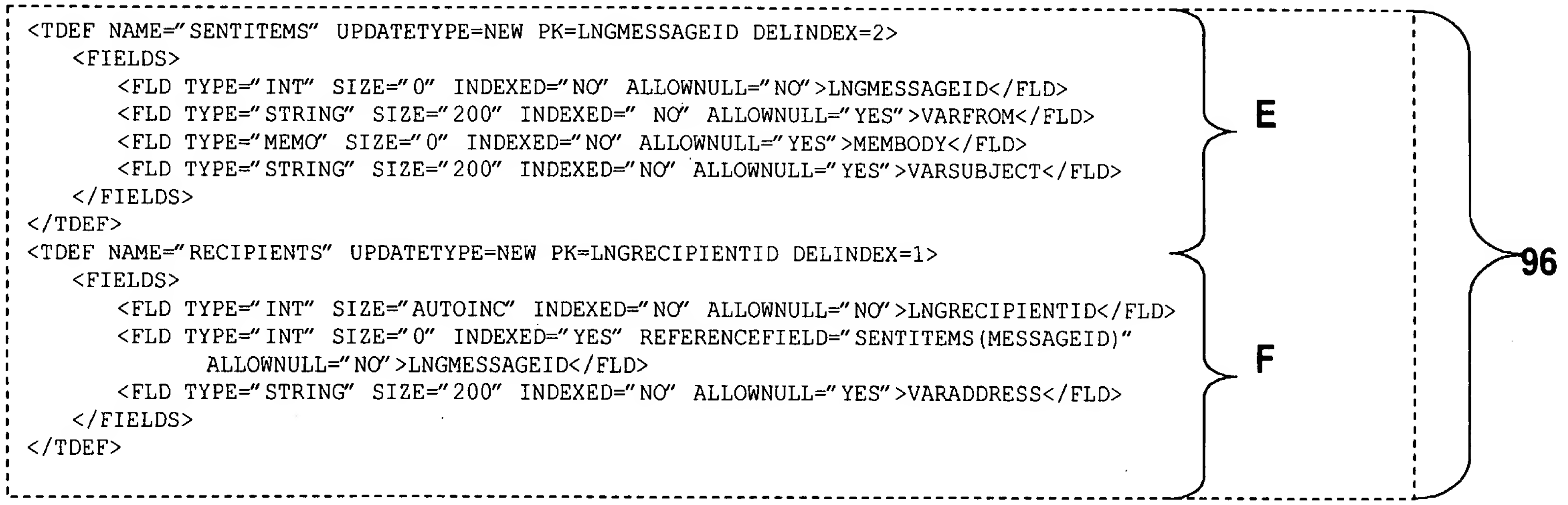
```
<ARML>
   <SCREEN NAME="NewMsg">
      <BUTTONS NUM="1">
         <BTN NAME="OK" CAPTION="Send" INDEX="0">
            <EVENTS NUM="1">
               <ACTION TYPE="ARML">
                  <ARMLTEXT>
                     <PKG TYPE="ME">
                        <MAIL MSGID="1" FROM="[Sys.UserName]"
                           SUBJECT="[NewMsg.Subject]">
                           <DATA>[NewMsg.Body]</DATA>
                           <RECIPS>
                              <RCP MSGID="1" TO="[NewMsg.To]"></RCP>
                           </RECIPS>
                        </ME>
                     </BODY>
                  </ARMLTEXT>
               </ACTION>
            </EVENTS>
         </BTN>
      </BUTTONS>
      <EDITBOXES NUM="3">
         <EB NAME="To" INDEX="1"></EB>
         <EB NAME="Subject" INDEX="2"></EB>
         <EB NAME="Body" INDEX="3"></EB>
      </EDITBOXES>
   </SCREEN>
</ARML>
```

Figure 14



```
<PKG TYPE="ME">
   <MAIL MSGID="1" FROM="Tim Neil" FROMADDRESS="timn@nextair.com" SUBJECT="Hello Back">
   <DATA>I am responding to your message</DATA>
   </MAIL>
   <RECIPS>
      <RCP MSGID="1" ADDRESS="steveh@nextair.com"></RCP>
   </RECIPS>
</PKG>
```

FIG. 15A



```
<TDEF NAME="SENTITEMS" UPDATETYPE=NEW PK=LNGMESSAGEID DELINDEX=2>
    <FIELDS>
        <FLD TYPE="INT" SIZE="0" INDEXED="NO" ALLOWNULL="NO">LNGMESSAGEID</FLD>
        <FLD TYPE="STRING" SIZE="200" INDEXED=" NO" ALLOWNULL="YES">VARFROM</FLD>
        <FLD TYPE="MEMO" SIZE="0" INDEXED="NO" ALLOWNULL="YES">MEMBODY</FLD>
        <FLD TYPE="STRING" SIZE="200" INDEXED="NO" ALLOWNULL="YES">VARSUBJECT</FLD>
    </FIELDS>
</TDEF>
<TDEF NAME="RECIPIENTS" UPDATETYPE=NEW PK=LNGRECIPIENTID DELINDEX=1>
    <FIELDS>
        <FLD TYPE="INT" SIZE="AUTOINC" INDEXED="NO" ALLOWNULL="NO">LNGRECIPIENTID</FLD>
        <FLD TYPE="INT" SIZE="0" INDEXED="YES" REFERENCEFIELD="SENTITEMS(MESSAGEID)"
                ALLOWNULL="NO">LNGMESSAGEID</FLD>
        <FLD TYPE="STRING" SIZE="200" INDEXED="NO" ALLOWNULL="YES">VARADDRESS</FLD>
    </FIELDS>
</TDEF>
```

FIG. 15B

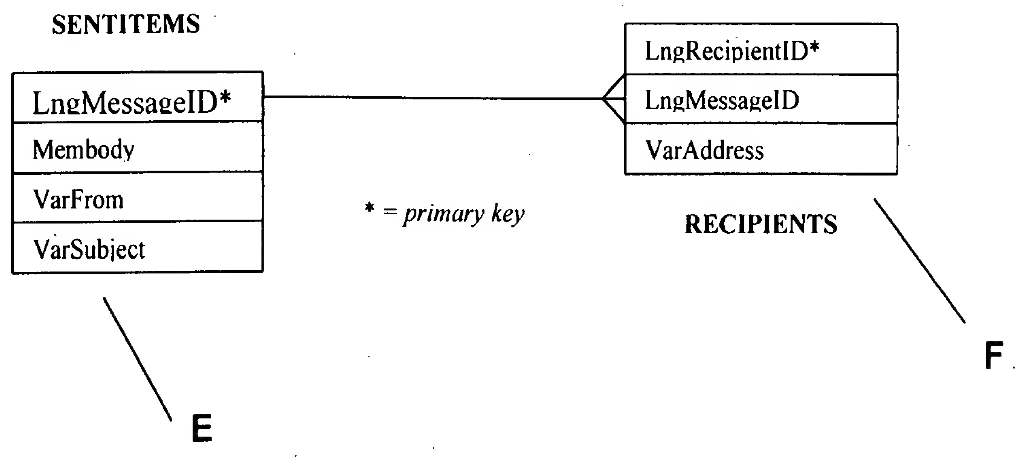


FIG. 15C

```
<AXDATAPACKET BODY="ME" SENDTOMOBILE="NO" SENDTOAPP="YES">
   <TABLEUPDATES>

      <TUPDATE TABLE="SENTITEMS" UPDATETYPE="ADD" WHERETYPE="PROP"
            SECTION="MAIL" MULTIROW="NO">
         <FIELDS>
            <PKGFLD NAME="LNGMESSAGEID" PARAMTYPE="PROP">MSGID</PKGFLD>
            <PKGFLD NAME="VARFROM" PARAMTYPE="PROP">FROM</PKGFLD>
            <PKGFLD NAME="VARSUBJECT" PARAMTYPE="PROP">SUBJECT</PKGFLD>
            <PKGFLD NAME="MEMBODY" PARAMTYPE="VALUE">DATA</PKGFLD>
         </FIELDS>
      </TUPDATE>

      <TUPDATE TABLE="RECIPIENTS" UPDATETYPE="ADD" WHERETYPE="PROP"
            SECTION="RECIPS" MULTIROW="YES" MULTIROWIDENT="RCP">
         <FIELDS>
            <PKGFLD NAME="LNGMESSAGEID" PARAMTYPE="PROP">MSGID</PKGFLD>
            <PKGFLD NAME="VARFULLNAME" PARAMTYPE="PROP">TO</PKGFLD>
            <PKGFLD NAME="VARADDRESS" PARAMTYPE="PROP">ADDRESS</PKGFLD>
         </FIELDS>
      </TUPDATE>

   </TABLEUPDATES>
</AXDATAPACKET>
```

108

104

106

Appendix "A"

ARML Language Specification

© 2000-2001, Nextair Corporation

FIG. 16A

Contents

FIG. 16B

FIG. 16C

FIG. 16D

1 Introduction

1.1 Purpose of document

This document describes the structure and syntax of the ARML language.

1.2 Audience

The document is intended to be read by AIRIX developers and users of ARML.

1.3 Definitions & Acronyms

ARML	AIRIX Markup Language
XML	Extensible Markup Language

FIG. 16E

2 ARML Overview

ARML is an XML markup language used by the AIRIX platform. It performs three tasks;

- Data is passed back and forth between the mobile server, AIRIX platform and enterprise application using ARML.
- The AIRIX Virtual machine uses ARML to define the user interface for an AIRIX-enabled application on the mobile device
- The AIRIX server uses ARML to define that data that it stores for the application in its database.

2.1 ARML design considerations

ARML has been designed with the following goals in mind;

- Transactions and screen definitions should be as independent as possible
- AIRIX should be unaware of internals of the enterprise application
- Strict conformance to the XML specification will be enforced
- Operation should be transparent to the end user
- ARML packages should be readable as is
- The minimum number of characters needed should be used

FIG. 16F

2.2 ARML usage

The diagram below illustrates how ARML is used.

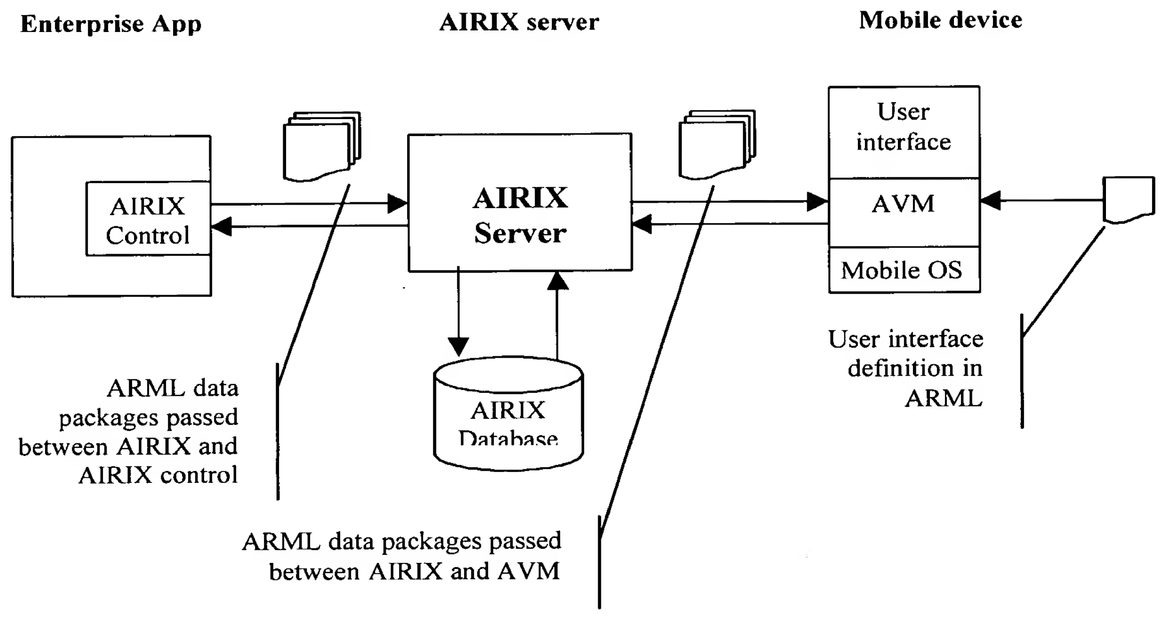

Enterprise App
AIRIX server
Mobile device
AIRIX Control
AIRIX Server
User interface
AVM
Mobile OS
AIRIX Database
ARML data packages passed between AIRIX and AIRIX control
ARML data packages passed between AIRIX and AVM
User interface definition in ARML

Figure 1 -The ARML environment

The key to ARML usage is the application definition file held on the AIRIX server. This file defines the AIRIX tables for the application, the allowed message set and the user interface definitions for the application on a given device.

2.3 The ARML prolog

As ARML is XML, all ARML documents must start with a prolog containing an XML declaration and a document type declaration, that precedes the actual ARML. The following prolog is appropriate;

```
<?xml version="1.0"?>
<!DOCTYPE ARML PUBLIC "-//NEXTAIR//DTD ARML 1.0//EN"
"http://www.nextair.com/DTD/ARML_1.0.xml">
```

2.4 The scratchpad area

Sometimes information needs to be passed from one screen to the next. This is achieved by the scratchpad, a temporary storage area where screens can store the values of field for use later on.

FIG. 16G

3 ARML application definition

3.1 General

3.1.1 Description

The application definition section defines the AIRIX tables and ARML data packages that are used for transactions involved with a specific application.

3.1.2 Structure

The ARML application definition has the following structure;

```
<ARML>
      <AXSCHDEF>
            <AXTDEFS>
                  (table definitions)
            </AXTDEFS>
            <DPACKETS>
                  (data package definitions)
            </DPACKETS>
            <DEVICES>
                  (device interface definitions)
            </DEVICES>
      </AXSCHDEF>
</ARML>
```

3.1.3 Tags

3.1.3.1 The <AXSCHDEF> tag

These tags (<AXSCHDEF>...</AXSCHDEF>) mark the start and end of the application definition. THE AXSCHDEF tag has two attributes;

Attribute	**Optional?**	**Description**
APPNAME	No	The name of the application
VERSION	No	Which version of the application the file describes

3.1.3.2 The <AXTDEFS> tag

The <AXTDEFS>...</AXTDEFS> pair marks the start and end of the table definitions section. It has no attributes.

3.1.3.3 The <DPACKETS> tag

The <DPACKETS>...</DPACKETS> pair marks the start and end of the data package definitions section. It has no attributes.

3.1.3.4 The <DEVICES> tag

The <DEVICES>...</DEVICES> pair marks the start and end of the device interface definitions section. It has no attributes.

FIG. 16H

3.2 Table Definitions Section

3.2.1 Description

The table definitions section defines the tables on the AIRIX server for the application

3.2.2 Structure

The table definitions section has the following structure;

```
{wrapper tags}
<TDEF>
      <FIELDS>
            <FLD>...</FLD>
      <FIELDS>
</TDEF>
 (etc.)
{wrapper tags}
```

3.2.3 Tags

3.2.3.1 The <TDEF> tag

Each table definition is enclosed within the <TDEF>...</TDEF> pair. The TDEF tag has the following attributes;

Attribute	Optional?	Description
NAME	No	The number of table definitions in the section
UPDATETYPE	No	Permitted values are: NEW –
PK	No	Which of the table fields is the primary key for the table

3.2.3.2 The <FIELDS> tag

The <FIELDS>...</FIELDS> tag pair marks where the fields in a given table are defined. The FIELDS tag has a no attributes.

3.2.3.3 The <FLD> tag

The <FLD>...</FLD> tag pair defines a single field in a table. Enclosed between the tags is the field name. The <FLD> tag has the following structure;

Attribute	Optional?	Description
TYPE	No	The data type contained in the field. Permitted values are: INT – integer value STRING – a fixed-length string of n characters (see SIZE field) MEMO – a string of max 65535 characters
SIZE	No	If the TYPE is set to STRING, this field specifies the number of characters in the field
INDEXED	No	Specifies if the field needs to be indexed in the AIRIX database
REFERENCEFIELD	Yes	
ALLOWNULL	No	Specifies if the field is allowed to have a null value

FIG. 16I

3.2.4 Example

An email application would use 2 tables for storing sent emails.

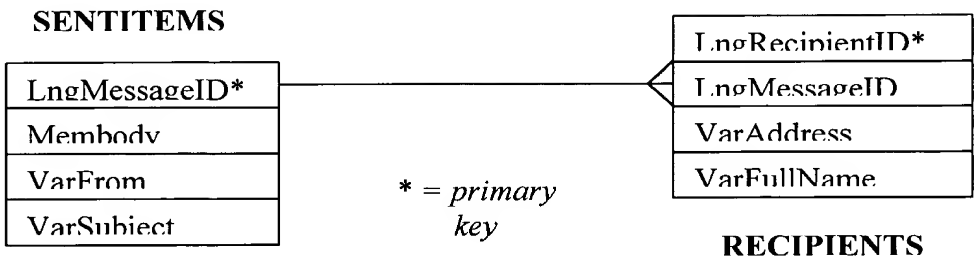


Figure 2 - sample email schema

This translates into the following ARML fragment;

```
<TDEF NAME="SENTITEMS" UPDATETYPE=NEW PK=LNGMESSAGEID>
   <FIELDS>
      <FLD TYPE="INT" SIZE="0" INDEXED="NO" REFERENCEFIELD=""
            ALLOWNULL="NO">LNGMESSAGEID</FLD>
      <FLD TYPE="STRING" SIZE="200" INDEXED="NO" REFERENCEFIELD=""
            ALLOWNULL="YES">VARFROM</FLD>
      <FLD TYPE="MEMO" SIZE="0" INDEXED="NO" REFERENCEFIELD=""
            ALLOWNULL="YES">MEMBODY</FLD>
      <FLD TYPE="STRING" SIZE="200" INDEXED="NO" REFERENCEFIELD=""
            ALLOWNULL="YES">VARSUBJECT</FLD>
   </FIELDS>
</TDEF>
<TDEF NAME="RECIPIENTS" UPDATETYPE=NEW PK=LNGRECIPIENTID>
   <FIELDS>
      <FLD TYPE="INT" SIZE="AUTOINC" INDEXED="NO" REFERENCEFIELD=""
            ALLOWNULL="NO">LNGMESSAGEID</FLD>
      <FLD TYPE="INT" SIZE="0" INDEXED="YES"
            REFERENCEFIELD="SENTITEMS(MESSAGEID)"
            ALLOWNULL="NO">LNGMESSAGEID</FLD>
      <FLD TYPE="STRING" SIZE="200" INDEXED="NO" REFERENCEFIELD=""
            ALLOWNULL="YES">VARFULLNAME</FLD>
      <FLD TYPE="STRING" SIZE="200" INDEXED="NO" REFERENCEFIELD=""
            ALLOWNULL="YES">VARADDRESS</FLD>
   </FIELDS>
</TDEF>
```

Figure 3 - a sample table definition section

FIG. 16J

3.3 Package Definitions Section

3.3.1 Description

The package definitions section defines the structure of the application packages and the data that they carry.

3.3.2 Structure

The package definitions section has the following structure;

```
{wrapper tags}
<AXDATAPACKET>
        <TABLEUPDATES>
                <TUPDATE>
                        <FIELDS>
                                <FLD>...</FLD>
                                <FLD>...</FLD>
                        <FIELDS>
                </TUPDATE>
        </TABLEUPDATES>
        <TABLEUPDATES>
                <TUPDATE>
                        <FIELDS>
                                <FLD>...</FLD>
                                <FLD>...</FLD>
                                (etc.)
                        <FIELDS>
                </TUPDATE>
        </TABLEUPDATES>
          (etc.)
</AXDATAPACKET>
{wrapper tags}
```

3.3.3 Tags

3.3.3.1 The <AXDATAPACKET> tag

The <AXDATAPACKET>...</AXDATAPACKET> pair delimits a package definition. The tag has the following attributes;

Attribute	Optional?	Description
BODY	No	This field gives the name by which the data package is known
SENDTOMOBILE	No	Specifies whether the package is sent to the mobile device
SENDTOAPP	No	Specifies whether the package is sent to the application server

3.3.3.2 The <TABLEUPDATES> tag

The <TABLEUPDATES>...</TABLEUPDATES> pair marks the start and end of the table definitions section. It has no attributes.

3.3.3.3 The <TUPDATE> tag

Each table update is enclosed within the <TUPDATE>...</TUPDATE> pair. The TUPDATE tag has the following attributes;

Attribute	Optional?	Description

FIG. 16K

TABLE	No	The table in the database that is updated
UPDATETYPE	No	
WHEREFIELD	Yes	
WHEREPARAM	Yes	
WHERETYPE	No	
SECTION	No	
MULTIROW	No	
MULTIROWINDENT	Yes	

3.3.3.4 The <PKGFIELDS> tag

The <PKGFIELDS>…</PKGFIELDS> tag pair marks where the fields in a given data package are defined. The PKGFIELDS tag has no attributes.

3.3.3.5 <The PKGFLD> tag

The <PKGFLD>…</PKGFLD> tag pair defines a single parameter in a given data package. Enclosed between the <PKGFLD>…</PKGFLD> tags is the field name. The <PKGFLD> tag has the following attributes;

Attribute	**Optional?**	**Description**
NAME	No	This is the field in the AIRIX database that maps to the user interface field
PARAMTYPE	No	This defines the type of parameter. It can take two values; PROP – this means that the parameter appears as part of the tag definition VALUE – this means that the parameter is contained between the two tags. Only one parameter in a given data package can be of this type

FIG. 16L

3.3.4 Example

Using the table definitions example in section 3.2.4, when the user sends an email, a data package to transport the data would update the 'SENTITEMS' table and the 'RECIPIENTS' table. The following ARML fragment defines such a data package;

```
<AXDATAPACKET BODY="ME" SENDTOMOBILE="NO" SENDTOAPP="YES">
    <TABLEUPDATES>
        <TUPDATE TABLE="SENTITEMS" UPDATETYPE="ADD" WHEREFIELD=""
        WHEREPARAM=""
                WHERETYPE="PROP" SECTION="MAIL" MULTIROW="NO" MULTIROWIDENT="">
            <FIELDS>
                <PKGFLD NAME="LNGMESSAGEID" PARAMTYPE="PROP">MSGID</PKGFLD>
                <PKGFLD NAME="VARFROM" PARAMTYPE="PROP">FROM</PKGFLD>
                <PKGFLD NAME="VARSUBJECT" PARAMTYPE="PROP">SUBJECT</PKGFLD>
                <PKGFLD NAME="MEMBODY" PARAMTYPE="VALUE">DATA</PKGFLD>
            </FIELDS>
        </TUPDATE>
        <TUPDATE TABLE="RECIPIENTS" UPDATETYPE="ADD" WHEREFIELD=""
        WHEREPARAM=""
                WHERETYPE="PROP" SECTION="RECIPS" MULTIROW="YES"
                MULTIROWIDENT="RCP">
            <FIELDS>
                <PKGFLD NAME="LNGMESSAGEID" PARAMTYPE="PROP">MSGID</PKGFLD>
                <PKGFLD NAME="VARFULLNAME" PARAMTYPE="PROP">TO</PKGFLD>
                <PKGFLD NAME="VARADDRESS" PARAMTYPE="PROP">ADDRESS</PKGFLD>
            </FIELDS>
        </TUPDATE>
    </TABLEUPDATES>
</AXDATAPACKET>
```

Figure 4 - a sample package definition

FIG. 16M

3.4 Device Interface Definitions Section

3.4.1 Description

The display definitions section contains the user interface definitions for the various mobile devices that an application supports.

3.4.2 Structure

The device display definitions section has the following structure;

```
{wrapper tags}
<DEV>
      <SCREENS>
            <SCRN>

            </SCRN>
      </SCREENS>
</DEV>
(other devices)
{wrapper tags}
```

3.4.3 Tags

3.4.3.1 The <DEV> tag

The <DEV>...</DEV> pair delimits an interface definition for a specific device. The tag has the following attributes;

Attribute	Optional?	Description
TYPE	No	The type of device. Allowed values are:

3.4.3.2 The <SCREENS> tag

The <SCREENS>...</SCREENS> pair delimits the screens definition for a specific device. The tag has the following attribute;

Attribute	Optional?	Description
LANGUAGE	No	The language that the screens grouping is in. This uses the IETF language identifiers as defined in RFC 1766.

3.4.3.3 The <SCRN> tag

The <SCRN>...</SCRN> pair delimits a screen definition. The tag pair contains the name of the file with the screen definition. The tag has the following attributes;

Attribute	Optional?	Description
NAME	No	An internal identifier for the screen

FIG. 16N

3.4.4 Example

The following example shows the screen definitions section for an application that allows a user to view their inbox and the mails in it.

```
{wrapper tags}
<DEV TYPE="RIM">
       <SCREENS LANGUAGE="EN">
              <SCRN NAME="INBOX.screen"></SCRN>
              <SCRN NAME="VIEWNEWMAIL.screen"></SCRN>
       </SCREENS>
</DEV>
<DEV TYPE="PALM">
       <SCREENS LANGUAGE="EN">
              <SCRN NAME="INBOX.screen"></SCRN>
              <SCRN NAME="VIEWNEWMAIL.screen"></SCRN>
       </SCREENS>
</DEV>
{wrapper tags}
```

FIG. 16O

4 Application-defined packages

This section describes the format of application defined packages.

4.1 General

This section describes the general structure of an application-specific data package. As described in section , ;

4.1.1 Description

System level packages are sent between AIRIX and the application server, and between AIRIX and the AVM

4.1.2 Structure

An application defined package has the following structure;

```
<ARML>
        <HEAD>
                (header information)
        </HEAD>
        <PKG>
                (package information)
        </PKG>
</ARML>
```

4.1.3 Tags

4.1.3.1 The <HEAD> tag

The <HEAD> tag is as described in section 6.1.3.1

4.1.3.2 The <PKG> tag

The <PKG>...</PKG> tags delimit the package data. The PKG tag has the following attributes;

Attribute	Optional?	Description
TYPE	No	A text string identifying the type of package being sent

FIG. 16P

4.2 Package information

The format and rules for application-defined data packages depend on the package definitions for that application.

4.2.1 Example

A sample data package following the rules in section 3.3.4 would have a body section like this;

```
{wrapper tags}
<PKG TYPE="ME">
    <MAIL MSGID="1" FROM="Tim Neil" FROMADDRESS="timn@nextair.com"
        SUBJECT="Hello Back">
    <DATA>I am responding to your message</DATA>
    </MAIL>
    <RECIPS>
        <RCP MSGID="1" TO="Jeff Jones"
            ADDRESS="jeff@nextair.com"></RCP>
        <RCP MSGID="1" TO="Scott Neil"
            ADDRESS="scottn@nextair.com"></RCP>
        <RCP MSGID="1" TO="Steve Hulaj"
            ADDRESS="steveh@nextair.com"></RCP>
    </RECIPS>
</PKG>
{wrapper tags}
```

Figure 5 - a sample package

We will use this sample package to illustrate how packages are derived from the package definition file. The first tag in the package is the BODY tag. This tag defines which type of package it is;

Package Definition

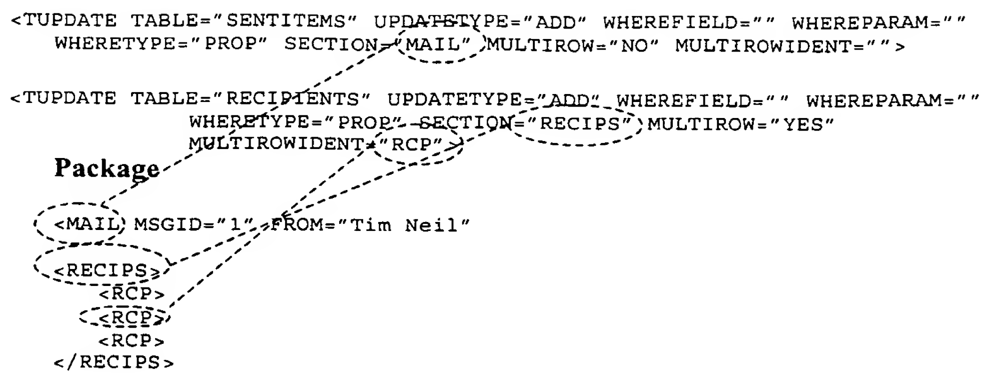
```
<AXDATAPACKET BODY="ME" SENDTOMOBILE="NO" SENDTOAPP="YES">
```

Package

```
<BODY TYPE="ME">
```

The package has two sections, which correspond to the two table update sections in the package definition;

FIG. 16Q



```
<TUPDATE TABLE="SENTITEMS" UPDATETYPE="ADD" WHEREFIELD="" WHEREPARAM=""
   WHERETYPE="PROP" SECTION="MAIL" MULTIROW="NO" MULTIROWIDENT="">

<TUPDATE TABLE="RECIPIENTS" UPDATETYPE="ADD" WHEREFIELD="" WHEREPARAM=""
             WHERETYPE="PROP" SECTION="RECIPS" MULTIROW="YES"
             MULTIROWIDENT="RCP">
```

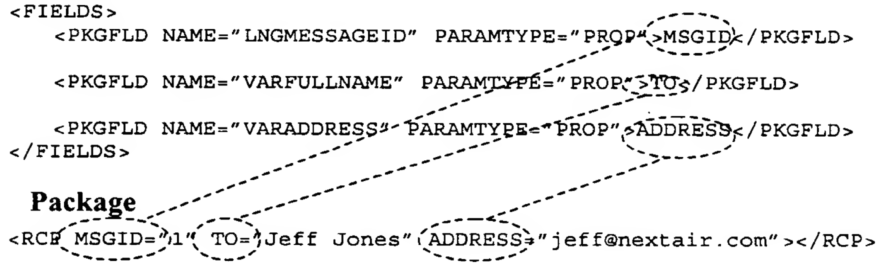
Package

```
<MAIL MSGID="1" FROM="Tim Neil"
<RECIPS>
   <RCP>
   <RCP>
   <RCP>
</RECIPS>
```

The 'MAIL' section updates the 'SENTITEMS' table in the database. It does not update multiple rows. The 'RECIPS' section updates the 'RECIPIENTS' table in the database; it does update multiple rows, and each row is contained within a pair of <RCP> tags.

Each of the MAIL and RCP tags have fields which are used to update the field in the database tables;

Package Definition



```
<FIELDS>
   <PKGFLD NAME="LNGMESSAGEID" PARAMTYPE="PROP">MSGID</PKGFLD>

   <PKGFLD NAME="VARFULLNAME" PARAMTYPE="PROP">TO</PKGFLD>

   <PKGFLD NAME="VARADDRESS" PARAMTYPE="PROP">ADDRESS</PKGFLD>
</FIELDS>
```

Package

```
<RCP MSGID="1" TO="Jeff Jones" ADDRESS="jeff@nextair.com"></RCP>
```

FIG. 16R

5 Screen Definitions

5.1 General

5.1.1 Description

A screen definition file defines a single screen for a specific device.

5.1.2 Structure

A screen definition file has the following structure;

```
<ARML>
      <SCREEN>
            <MENU>
                  (menu definition)
            </MENU>
            <BUTTONS>
                  (button definitions)
            </BUTTONS>
            <TEXTITEMS>
                  (textitem definitions)
            </TEXTITEMS>
            <EDITBOXES>
                  (edit box definitions)
            </EDITBOXES>
            <CHOICEITEMS>
                  (choice item definitions)
            </CHOICEITEMS>
            <MESSAGEBOXES>
                  (message box definitions)
            </MESSAGEBOXES>
            <IMAGES>
                  (image definitions)
            </IMAGES>
            <LISTBOXES>
                  (list box definitions)
            </LISTBOXES>
            <CHECKBOXES>
                  (check box definitions)
            </CHECKBOXES>
      </SCREEN>
</ARML>
```

5.1.3 Tags

5.1.3.1 The SCREEN tag

The <SCREEN>...</SCREEN> pair marks the start and end of the screen definitions section. It has attribute –

Attribute	Optional?	Description
NAME	No	An identifier for the screen. This is used to qualify variables and navigate between screens
TITLE	No	The title that appears for the screen.
BACKGROUND	Yes	If used, an image that appears behind the interface elements

FIG. 16S

5.1.3.2 The BUTTONS tag

The <BUTTONS>...</BUTTONS> pair marks the start and end of the screen definitions section. It has no attributes.

5.1.3.3 The TEXTITEMS tag

The <TEXTITEMS>...</TEXTITEMS> pair marks the start and end of the text items section. It has no attributes.

5.1.3.4 The EDITBOXES tag

The <EDITBOXES>...</EDITBOXES> pair marks the start and end of the editboxes section. It has no attributes.

5.1.3.5 The CHOICEITEMS tag

The <CHOICEITEMS>...</CHOICEITEMS> pair marks the start and end of the images section. It has no attributes.

5.1.3.6 The MESSAGEBOXES tag

The <MESSAGEBOXES>...</MESSAGEBOXES> pair marks the start and end of the checkboxes section. It has no attributes.

5.1.3.7 The IMAGES tag

The <IMAGES>...</IMAGES> pair marks the start and end of the images section. It has no attributes.

5.1.3.8 The CHECKBOXES tag

The <CHECKBOXES>...</CHECKBOXES> pair marks the start and end of the checkboxes section. It has no attributes.

5.1.3.9 The LISTBOXES tag

The <LISTBOXES>...</LISTBOXES> pair marks the start and end of the listboxes section. It has no attributes.

5.2 Menu definition section

5.2.1 Description

The menu definition section describes the menu for a given screen.

5.2.2 Structure

The menu definition section has the following structure;

```
{wrapper tags}
<MENU>
      <MENUITEM>
            <EVENTS>
                  <ACTION>…</ACTION>
            </EVENTS>
      </MENUITEM>
</MENU>
```

FIG. 16T

```
{wrapper tags}
```

5.2.3 Tags

5.2.3.1 The EVENTS tag

The <EVENTS>...</EVENTS> pair marks the start and end of the events section. It has no attributes.

5.2.3.2 The ACTION tag

The <ACTION>...</ACTION> pair marks the start and end of an event definition. It has attribute –

Attribute	Optional?	Description
EVENTTYPE		The type of action that should be performed when the button is pushed. Allowed values are; OPEN – tells the AVM to open the screen with the name given ARML – tells the AVM to compose & send an ARML package to the server using info derived from fields on the screen SAVE – tells the AVM to cache all fields that are marked as needed to be saved in the scratchpad area

5.3 Buttons definition section

5.3.1 Description

The buttons definition section describes the buttons that appear on a given screen.

5.3.2 Structure

The buttons definition section has the following structure;

```
{wrapper tags}
<BTN>
        <EVENTS>
                <ACTION>...</ACTION>
        </EVENTS>
</BTN>
{wrapper tags}
```

5.3.3 Tags

5.3.3.1 The BTN tag

The <BTN>...</BTN> pair marks the start and end of a button definition. It has one attribute –

Attribute	Optional?	Description
NAME	No	An identifier for the button.
INDEX	No	The order in which the button appears
CAPTION	No	The caption that appears on a given button
X	Yes	The X-coordinate of the button on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
Y	Yes	The Y-coordinate of the button on the screen. This attribute may not be meaningful in some display environments, in which case it would be

FIG. 16U

		skipped without processing by the parser

5.3.3.2 The EVENTS tag

The events tag is as in section 5.2.3.1

5.3.3.3 The ACTION tag

The action tag is as in section 5.2.3.2

5.4 Text Items definition section

5.4.1 Description

The text items definition

5.4.2 Structure

The text items section has the following structure;

```
{wrapper tags}
<TI>
      <EVENTS>
            <ACTION>…</ACTION>
      </EVENTS>
</TI>
{wrapper tags}
```

5.4.3 Tags

5.4.3.1 The TI tag

The <TI>…</TI> pair marks the start and end of the screen definitions section. It has attribute –

Attribute	Optional?	Description
INDEX	No	The order in which the text item appears
X	Yes	The X-coordinate of the text item on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
Y	Yes	The Y-coordinate of the text item on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser

5.5 Edit boxes definition section

5.5.1 Description

5.5.2 Structure

The edit boxes section has the following structure;

```
{wrapper tags}
<EB>
      <EVENTS>
            <ACTION>…</ACTION>
      </EVENTS>
</EB>
{wrapper tags}
```

FIG. 16V

5.5.3 Tags

5.5.3.1 The EB tag

The <EB>...</EB> pair marks an edit box definition. It has the following attributes –

Attribute	Optional?	Description
NAME	No	An identifier for the edit box.
INDEX	No	The order in which the edit box appears
CAPTION	No	The caption for on a given edit box
MULTILINE	No	Boolean field that indicates whether the edit box is a multiline field.
Save	No	Boolean value indicating whether or not to save the value in this field to temporary storage for use by other screens later on. Saving the value to the scratchpad is triggered by either exiting the screen or by an explicit 'SAVE' action on a user interface control.
X	Yes	The X-coordinate of the edit box on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
Y	Yes	The Y-coordinate of the edit box on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
DATASRC	Yes	If present, the package and field in the package that populates this edit box. This is given in the format "package.field".

5.5.3.2 The EVENTS tag

The events tag is as described in section 5.2.3.1

5.5.3.3 The ACTION tag

The action tag is as described in section 5.2.3.2

5.6 Choice items definition section

5.6.1 Description

The choice item definitions section describes the choice items that exist on a given screen. A choice item is an interface item that requires the user to make a selection from a list of options. It can be represented in different ways on different devices; on a RIM pager, it is a choice box, while on a WinCE device, it is a drop-down list.

5.6.2 Structure

The choice items section has the following structure;

```
{wrapper tags}
<CHOICE>
        <EVENTS>
                <ACTION>...</ACTION>
        </EVENTS>
</CHOICE>
{wrapper tags}
```

FIG. 16W

5.6.3 Tags

5.6.3.1 The <CHOICE> tag

The <CHOICE>…</CHOICE> pair marks the start and end of a choice item definition. It has these attributes –

Attribute	Optional?	Description
NAME	No	An identifier for the choice item.
INDEX	No	The order in which the choice item appears
CAPTION	No	The caption that appears for a given choice item
Save	No	Boolean value indicating whether or not to save the value in this field to temporary storage for use by other screens later on. Saving the value to the scratchpad is triggered by either exiting the screen or by an explicit 'SAVE' action on a user interface control.
X	Yes	The X-coordinate of the choice item on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
Y	Yes	The Y-coordinate of the choice item on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
DATASRC	Yes	If present, the package and field in the package that populates this choice item. This is given in the format "package.field".

5.7 Messageboxes definition section

5.7.1 Description

The messageboxes section describes the messageboxes that could appear due to user action.

5.7.2 Structure

The messageboxes section has the following structure;

```
{wrapper tags}
<MB>
       <EVENTS>
              <ACTION>...</ACTION>
       </EVENTS>
</MB>
{wrapper tags}
```

5.7.3 Tags

5.7.3.1 The MB tag

The <MB>…</MB> pair marks a message box definition

Attribute	Optional?	Description
CAPTION	Yes	The caption to display in the title bar of the message box
TEXT	Yes	The text to display in the message box
TYPE	No	The type of message box to display

FIG. 16X

5.8 Images definition section

5.8.1 Description

The images section describes.

5.8.2 Structure

The messageboxes section has the following structure;

```
{wrapper tags}
        <IMG>...</IMG>
{wrapper tags}
```

5.8.3 Tags

5.8.3.1 The IMG tag

The <IMG>…</IMG> pair describes an image that appears on a given screen.

Attribute	Optional?	Description
NAME	No	An identifier for the image.
FILE	No	The filename of the image.
X	Yes	The X-coordinate of the image on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
Y	Yes	The Y-coordinate of the image on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser

5.9 Listboxes definition section

5.9.1 Description

The listboxes section describes a list box that appears on a given screen.

5.9.2 Structure

The listboxes section has the following structure;

```
{wrapper tags}
        <LB>...</LB>
{wrapper tags}
```

5.9.3 Tags

5.9.3.1 The LB tag

The <LB>…</LB> pair marks a list box definition

Attribute	Optional?	Description
NAME	No	An identifier for the list box.
SAVE	No	Boolean value indicating whether or not to save the value in this field to temporary storage for use by other screens later on. Saving the value to the scratchpad is triggered by either exiting the screen or by an explicit 'SAVE' action on a user interface control.
X	Yes	The X-coordinate of the list box on the screen. This attribute may not be

FIG. 16Y

		meaningful in some display environments, in which case it would be skipped without processing by the parser
Y	Yes	The Y-coordinate of the list box on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
DATASRC	Yes	If present, the package and field in the package that populates this list box. This is given in the format "package.field".

5.10 Checkboxes definition section

5.10.1 Description

The checkboxes section describes a check box that appears on a given screen.

5.10.2 Structure

The checkboxes section has the following structure;

```
{wrapper tags}
      <CHK>...</CHK>
{wrapper tags}
```

5.10.3 Tags

5.10.3.1 The CHK tag

The <CHK>...</CHK> pair marks a check box definition

Attribute	Optional?	Description
NAME	No	An identifier for the check box.
Save	No	Boolean value indicating whether or not to save the value in this field to temporary storage for use by other screens later on. Saving the value to the scratchpad is triggered by either exiting the screen or by an explicit 'SAVE' action on a user interface control.
X	Yes	The X-coordinate of the check box on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
Y	Yes	The Y-coordinate of the check box on the screen. This attribute may not be meaningful in some display environments, in which case it would be skipped without processing by the parser
DATASRC	Yes	If present, the package and field in the package that populates this check box. This is given in the format "package.field".

5.11 Example of screen usage

The following example serves to illustrate how a screen is used to compose a data package to be sent back to the AIRIX server. The example used is a screen giving the bare functionality for composing a basic email message – to simplify the example, the user cannot cancel the action, and multiple recipients are not allowed.

```
<ARML>
   <SCREEN NAME="NewMsg">
      <BUTTONS>
         <BTN NAME="OK" CAPTION="Send" INDEX="0">
```

FIG. 16Z

```
            <EVENTS>
                <ACTION TYPE="ARML">
                    <ARMLTEXT>
                        <BODY TYPE="ME">
                            <ME MSGID="1" FROM="Tim Neil"
                                SUBJECT="[NewMsg.Subject]">
                                <DATA>[NewMsg.Body]</DATA>
                                <RECIPS>
                                    <RCP MSGID="1" TO="[NewMsg.To]"></RCP>
                                </RECIPS>
                            </ME>
                        </BODY>
                    </ARMLTEXT>
                </ACTION>
            </EVENTS>
        </BTN>
    </BUTTONS>
    <EDITBOXES>
        <EB NAME="To" INDEX="1"></EB>
        <EB NAME="Subject" INDEX="2"></EB>
        <EB NAME="Body" INDEX="3"></EB>
    </EDITBOXES>
  </SCREEN>
</ARML>
```

The Editboxes section at the bottom defines 3 editboxes, with the names of 'To', 'Subject', and 'Body';

```
        <EB NAME="To" INDEX="1"></EB>
        <EB NAME="Subject" INDEX="2"></EB>
        <EB NAME="Body" INDEX="3"></EB>
```

There is one button on the screen, with the name of 'OK';

```
        <BTN NAME="OK" CAPTION="Send" INDEX="0">
```

When the user clicks on OK, the button composes an ARML package to be sent to the AIRIX server;

```
            <EVENTS>
                <ACTION TYPE="ARML">
```

The ARML package sent is an 'ME' package as described in the example in section 4.2.1. It is composed as follows;

```
                        <BODY TYPE="ME">
                            <ME MSGID="1" FROM="Tim Neil"
                                SUBJECT="[NewMsg.Subject]">
                                <DATA>[NewMsg.Body]</DATA>
                                <RECIPS>
                                    <RCP MSGID="1" TO="[NewMsg.To]"></RCP>
                                </RECIPS>
                            </ME>
                        </BODY>
```

FIG. 16AA

The subject field is taken from the edit box named 'Subject';

```
<ME MSGID="1" FROM="Tim Neil" SUBJECT="[NewMsg.Subject]">
```

The recipients field is taken from the edit box named 'Subject';

```
<RECIPS>
   <RCP MSGID="1" TO="[NewMsg.To]"></RCP>
</RECIPS>
```

Finally the text of the message is filled from the 'Body' field;

```
<DATA>[NewMsg.Body]</DATA>
```

FIG. 16BB

6 System-level interactions

This section describes the primitives that are used for system-level interactions with the AIRIX server.

6.1 General

6.1.1 Description

System level packages are sent between AIRIX and the application server, and between AIRIX and the AVM

6.1.2 Structure

System interactions are performed by exchanging ARML data packages with the following structure;

```
<ARML>
<HEAD>...</HEAD>
<SYS>
{data}
</SYS>
</ARML>
```

6.1.3 Tags

6.1.3.1 The <HEAD> tag

The package header is delimited by the <HEAD>…</HEAD> tags. Contained in text between the two tags is the id of the destination mobile. The HEAD tag has the following attributes;

Attribute	Optional?	Description
DT	No	The date & time in RFC 1123 format
ID	No	A unique ID for the message
VERSION	No	The version number of the application
APPNAME	No	The application name
DEVICE	No	A numeric constant identifying the device

6.1.3.2 The <SYS> tag

The <SYS>…</SYS> pair contains the actual system package. The tag does not have any attributes.

FIG. 16CC

6.2 Device Registration & deregistration package

6.2.1 Description

Device registration packages are sent from the AIRIX component to the AIRIX server when a user changes their registration status.

6.2.2 Structure

A device registration package has the following structure;

```
{wrapper tags}
<REG>
      <CLIENTID> {data} </CLIENTID>
      <MOBILEID> {data} </MOBILEID>
      <NEWMOBILEID> {data} </NEWMOBILEID>
      <PLATFORMS>
            <PLATFORM> {data} </PLATFORM>
      </PLATFORMS>
</REG>
{wrapper tags}
```

6.2.3 Tags

6.2.3.1 The <REG> tag

The <REG>...</REG> pair delimit the registration request. The tag has the following attributes;

Attribute	Optional?	Description
TYPE	No	This defines the type of parameter. It can take two values; ADD – this means that the device is to be added to the registration database UPDATE – this means that the setting is being modified for the device DELETE – this means that the device is to be removed to the registration database
UPDATEPLATFORM	No	This field indicates whether the server will be updated. Allowable values are YES or NO

6.2.3.2 The <CLIENTID> tag

The <CLIENTID>...</CLIENTID> pair contain the clientID. The tag does not have any attributes.

6.2.3.3 The <MOBILEID> tag

The <MOBILEID>...</MOBILEID> pair contain the mobile ID. The tag does not have any attributes.

6.2.3.4 The <NEWMOBILEID> tag

The <MOBILEID>...</MOBILEID> pair contain the new mobileID. The tag does not have any attributes.

FIG. 16DD

6.2.3.5 The <PLATFORMS> tag

The <PLATFORMS>...</PLATFORMS> pair contain one or more PLATFORM declarations. The tag does not have any attributes.

6.2.3.6 The <PLATFORM> tag

The <PLATFORM>...</PLATFORM> pair contain the address to use for the platform. The tag has the following attributes;

Attribute	Optional?	Description
ID	No	The ID of the platform

6.2.4 Example

This package would be sent by a user, whose and who was going to use the RIM platform, to register;

```
{wrapper tags}
<REG TYPE="ADD">
       <CLIENTID>SUNTRESS</CLIENTID>
       <MOBILEID>867452</MOBILEID>
       <NEWMOBILEID>268625</NEWMOBILEID>
       <PLATFORMS>
              <PLATFORM>RIM</PLATFORM>
       </PLATFORMS>
</REG>
{wrapper tags}
```

FIG. 16EE

6.3 Registration confirmation package

6.3.1 Description

This packages is sent back from the AIRIX server to the AVM to confirm that the device has been registered.

6.3.2 Structure

A registration confirmation package has the following structure;

```
{wrapper tags}
<REGCONFIRM>
       <MOBILEID> {data} </MOBILEID>
       <VALUE> {data} </VALUE>
       <INTERFACE>
       {interface description}
       </INTERFACE>
</REGCONFIRM>
{wrapper tags}
```

6.3.3 Tags

6.3.3.1 The <REGCONFIRM> tag

The <REGCONFIRM>...</REGCONFIRM> pair delimit the confirmation. The tag has the following attributes;

Attribute	Optional?	Description
TYPE	No	This defines the type of parameter. It can take two values; ADD – this means that the device is to be added to the registration database UPDATE – this means that the setting is being modified for the device DELETE – this means that the device is to be removed to the registration database

6.3.3.2 The <MOBILEID> tag

The <MOBILEID>...</MOBILEID> pair contains the mobile ID. The tag does not have any attributes.

6.3.3.3 The <VALUE> tag

The <VALUE>...</VALUE> pair contains the status of the registration request. The following text strings are allowable;

CONFIRM – this means that the registration request was successful
EXCEEDLIMIT – this means that the registration request failed because
NOTUNIQUE – this means that the registration request failed because
INVALIDCLIENT – this means that the registration request failed because
NODEVICE – this means that the registration request failed because

FIG. 16FF

6.3.3.4 The <INTERFACE> tag

The <INTERFACE>...</INTERFACE> pair contains the user interface for the application. Th specification of the interface is as described in section 5;

6.3.4 Example

This package would be sent to confirm the example request in section 6.2.4;

```
{wrapper tags}
<REGCONFIRM TYPE="ADD">
       <MOBILEID>268625</MOBILEID>
       <VALUE>CONFIRM</VALUE>
       <INTERFACE>
           <BUTTONS NUM="1">
               <BTN NAME="OK" CAPTION="Send" INDEX="0">
               </BTN>
           </BUTTONS>
           <EDITBOXES NUM="3">
               <EB NAME="To" INDEX="1"></EB>
               <EB NAME="Subject" INDEX="2"></EB>
               <EB NAME="Body" INDEX="3"></EB>
           </EDITBOXES>
       </INTERFACE>
</REGCONFIRM>
{wrapper tags}
```

6.4 Setting the active device package

6.4.1 Description

If a user wishes to set the current device as their active device, the AVM must send a 'set active device' package to the AIRIX server

6.4.2 Structure

A 'set active device' package has the following structure;

```
{wrapper tags}
<SA>
{data}
</SA>
{wrapper tags}
```

6.4.3 Tags

6.4.3.1 The <SA> tag

The 'set active device' package is shown by the <SA>...</SA> tags. The tag has no attributes; the tag pair contains the user's username

6.4.4 Example

This package would be sent by a user with the username of 'scotty';

```
{wrapper tags}
<SA>scotty</SA>
{wrapper tags}
```

FIG. 16GG

6.5 Set active device response

6.5.1 Description

This packages is sent back from the AIRIX server to the AVM in response to a request to set the current device as the active one.

6.5.2 Structure

A 'set active device response' package has the following structure;

```
{wrapper tags}
<SACONFIRM>
       <VALUE> {data} </VALUE>
</SACONFIRM>
{wrapper tags}
```

6.5.3 Tags

6.5.3.1 The <SACONFIRM> tag

The <SACONFIRM>...</SACONFIRM> pair delimit the confirmation. The tag does not have any attributes.

6.5.3.2 The <VALUE> tag

The <VALUE>...</VALUE> pair contains the status of the registration request. The following text strings are allowable;

CONFIRM – this means that the registration request was successful
NOTREGISTERED – this means that the registration request failed because

6.5.4 Example

This package would be sent by the AIRIX server to confirm a set active request;

```
{wrapper tags}
<SACONFIRM>
       <VALUE>CONFIRM</VALUE>
</SACONFIRM>
{wrapper tags}
```

6.6 Set active platform package

6.6.1 Description

'Set active platform' packages are sent from the application to the AIRIX server to indicate that a particular device should be used for that application

6.6.2 Structure

A device registration package has the following structure;

```
{wrapper tags}
<SETPLATFORM>
       <CLIENTID> {data} </CLIENTID>
       <MOBILEID> {data} </MOBILEID>
       <PLATFORM> {data} </PLATFORM>
</SETPLATFORM>
```

FIG. 16HH

```
{wrapper tags}
```

6.6.3 Tags

6.6.3.1 The <SETPLATFORM> tag

The <SETPLATFORM>...</SETPLATFORM> pair delimit the registration request. The tag does not have any attributes.

6.6.3.2 The <CLIENTID> tag

The <CLIENTID>...</CLIENTID> pair contain the clientID. The tag does not have any attributes.

6.6.3.3 The <MOBILEID> tag

The <MOBILEID>...</MOBILEID> pair contains the mobile ID. The tag does not have any attributes.

6.6.3.4 The <PLATFORM> tag

The <PLATFORM>...</PLATFORM> pair contains the new mobileID. The tag does not have any attributes.

6.6.4 Example

This package would be sent by a user with the username of 'scotty';

```
{wrapper tags}
<SETPLATFORM TYPE="UPDATE">
       <CLIENTID>DEREKC</CLIENTID>
       <MOBILEID>102030</MOBILEID>
       <PLATFORM>WINCE</PLATFORM>
</SETPLATFORM>
{wrapper tags}
```

6.7 Set active platform response package

6.7.1 Description

This packages is sent back from the AIRIX server to the AVM in response to a request to set the current device as the active one.

6.7.2 Structure

A 'set active device response' package has the following structure;

```
{wrapper tags}
<PLATFORMCONFIRM>
       <MOBILEID> {data} </MOBILEID>
       <VALUE> {data} </VALUE>
</PLATFORMCONFIRM>
{wrapper tags}
```

FIG. 16II

6.7.3 Tags

6.7.3.1 The <PLATFORMCONFIRM> tag

The <PLATFORMCONFIRM>...</PLATFORMCONFIRM> pair delimit the confirmation. The tag has the following attributes;

Attribute	Optional?	Description
TYPE	No	This defines the type of parameter. It can take two values; ADD – this means that the device is to be added to the registration database UPDATE – this means that the setting is being modified for the device DELETE – this means that the device is to be removed to the registration database

6.7.3.2 The <MOBILEID> tag

The <MOBILEID>...</MOBILEID> pair contains the mobile ID. The tag does not have any attributes.

6.7.3.3 The <VALUE> tag

The <VALUE>...</VALUE> pair contains the status of the registration request. The following text strings are allowable;

CONFIRM – this means that the registration request was successful
NOTREGISTERED – this means that the registration request failed because
INVALIDCLIENT – this means that the registration request failed because
NODEVICE – this means that the registration request failed because
NETNOTREGISTERED – this means that the registration request failed because

6.7.4 Example

This package would be sent in response to the request in section 6.6.4 to indicate a failure;

```
{wrapper tags}
<PLATFORMCONFIRM TYPE="UPDATE">
       <MOBILEID>102030</MOBILEID>
       <VALUE>NOTREGISTERED</VALUE>
</PLATFORMCONFIRM>
{wrapper tags}
```

FIG. 16JJ